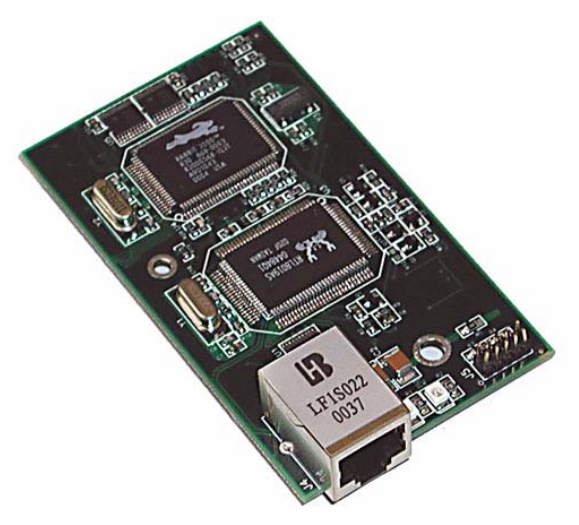



# RabbitCore RCM2100

C-Programmable Module with Ethernet

## Getting Started Manual

019–0093 • 050505–F

# RabbitCore RCM2100 Getting Started Manual

Part Number 019-0093 • 050505–F • Printed in U.S.A.

©2001–2005 Z-World Inc. • All rights reserved.

Z-World reserves the right to make changes and
improvements to its products without providing notice.

## Trademarks

Rabbit is a registered trademark of Rabbit Semiconductor.

Rabbit 2000 and RabbitCore are trademarks of Rabbit Semiconductor.

Z-World is a registered trademark of Z-World Inc.

Dynamic C is a registered trademark of Z-World Inc.

### Z-World, Inc.

2900 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-3737
Fax: (530) 757-3792

www.zworld.com

### Rabbit Semiconductor

2932 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-8400
Fax: (530) 757-8402

www.rabbitsemiconductor.com

# TABLE OF CONTENTS


**Chapter 1. Introduction & Overview** 1

1.1 RCM2100 Description ..... 1
1.1.1 Standard Ethernet Versions ..... 2
1.1.2 Standard Non-Ethernet Versions ..... 2
1.1.3 Physical & Electrical Specifications ..... 2
1.2 Development Software ..... 4
1.3 How to Use This Manual ..... 5
1.3.1 Additional Product Information ..... 5
1.3.2 Additional Reference Information ..... 5
1.3.3 Using Online Documentation ..... 5

**Chapter 2. Getting Started** 7

2.1 Development Kit Contents ..... 7
2.2 Overview of the Prototyping Board ..... 8
2.2.1 Prototyping Board Features ..... 9
2.2.2 Prototyping Board Expansion ..... 10
2.3 Connections ..... 11
2.3.1 Attach Module to Prototyping Board ..... 12
2.3.2 Connect Programming Cable ..... 13
2.3.3 Connect Power ..... 14
2.4 Run a Sample Program ..... 15
2.4.1 Troubleshooting ..... 15
2.5 Where Do I Go From Here? ..... 16
2.5.1 Technical Support ..... 16

**Chapter 3. Software Installation & Overview** **17**

3.1 An Overview of Dynamic C ..... 17
3.2 Installing Dynamic C ..... 19
3.2.1 Early Versions of Dynamic C ..... 19
3.3 Sample Programs ..... 20
3.3.1 Getting to Know the RCM2100 ..... 21
3.3.2 Serial Communication ..... 24
3.3.3 Other Sample Programs ..... 25
3.3.4 Sample Program Descriptions ..... 26
3.4 Upgrading Dynamic C ..... 29
3.4.1 Add-On Modules ..... 29

**Chapter 4. Using the TCP/IP Features** **31**

4.1 TCP/IP Connections ..... 31
4.2 TCP/IP Primer on IP Addresses ..... 33
4.3 IP Addresses Explained ..... 35
4.4 How IP Addresses are Used ..... 36
4.5 Dynamically Assigned Internet Addresses ..... 37
4.6 Placing Your Device on the Network ..... 38
4.7 Running TCP/IP Sample Programs ..... 39
4.8 How to Set IP Addresses in the Sample Programs ..... 40
4.9 How to Set Up your Computer's IP Address for Direct Connect ..... 41
4.10 Run the PINGME.C Sample Program ..... 42
4.11 Running More Sample Programs With Direct Connect ..... 42
4.11.1 Sample Program: PINGLED.C ..... 42
4.11.2 Sample Program: ETHCORE1.C ..... 44
4.11.3 Additional Sample Programs ..... 45
4.11.4 More Information ..... 45
4.12 Where Do I Go From Here? ..... 46

**Notice to Users** **47**

**Index** **49**

**Schematics** **51**

# 1. INTRODUCTION & OVERVIEW

The RCM2100 series is an advanced line of modules that incorporates the powerful Rabbit 2000® microprocessor, flash memory, static RAM and an RJ-45 Ethernet port, all on a PCB the size of a business card.

Throughout this manual, the term RCM2100 refers to the complete series of RCM2100 RabbitCore modules unless other production models are referred to specifically.

The RCM2100 modules are designed for use on a motherboard that supplies power and interface to real-world I/O devices. Up to 40 pins of I/O and four serial ports are available for system interfacing.

To accommodate a variety of user and production needs, the RCM2100 family includes versions with varying amounts of onboard memory. Models with and without the Ethernet port are available, to permit simultaneous development of Ethernet-capable and cheaper non-Ethernet versions of production systems. All modules within the family are pin-for-pin compatible and may be installed or swapped in a matter of minutes.

## 1.1 RCM2100 Description

There are four production models in the RCM2100 series. Their standard features are summarized in Table 1.

***Table 1. RCM2100 Versions***

| Feature | RCM2100 | RCM2110 | RCM2120 | RCM2130 |
|---|---|---|---|---|
| Microprocessor | Rabbit 2000 running at 22.1 MHz | | | |
| Flash Memory | 512K | 256K | 512K | 256K |
| Static RAM | 512K | 128K | 512K | 128K |
| General-Purpose I/O | 34 | | 40 | |
| Ethernet | 10/100-compatible<br>10Base-T interface | | None | |
| Serial Ports | 4, high-speed, CMOS-compatible;<br>2 configurable as clocked ports;<br>1 clocked port dedicated to programming port use. | | | |

### 1.1.1 Standard Ethernet Versions

There are two RCM2100 models that incorporate an RJ-45 Ethernet port:

**RCM2100.** The RCM2100 is the most fully equipped module in the family, with an Ethernet port, 512K flash memory, and 512K static RAM. The Ethernet port uses some of the Rabbit 2000 microprocessor's parallel ports, reducing the available number of I/O pins to 34. This is the model included in the Development Kit.

**RCM2110.** The RCM2110 is identical to the RCM2100 except that it is equipped with 128K SRAM and 256K flash memory.

### 1.1.2 Standard Non-Ethernet Versions

To accommodate developers and users who want the RCM2100's footprint and capabilities without the integrated Ethernet port, two standard versions of the module are available without the Ethernet hardware:

**RCM2120.** The RCM2120 is equipped with 512K flash memory and 512K static RAM, but does not include the Ethernet port hardware. In its place, ports D and E of the Rabbit 2000 microprocessor are enabled, giving this module 40 I/O pins.

**RCM2130.** The RCM2130 is identical to the RCM2120 except that it is equipped with 128K SRAM and 256K flash memory.

### 1.1.3 Physical & Electrical Specifications

Table 2 lists the basic specifications for the RCM2100 models.

***Table 2. RCM2100 Specifications***

| Specification | Data |
|---|---|
| Power Supply | 4.75–5.25 V DC (140 mA at 22.1 MHz clock speed) |
| Size | 2.0" × 3.5" × 0.85" (51mm × 89 mm × 22 mm) |
| Environmental | –40°C to 70°C, 5–95% humidity, noncondensing |

**NOTE:** For complete product specifications, see Appendix A in the ***RabbitCore RCM2100 User's Manual***.

The RCM2100 modules have two 40-pin headers to which cables can be connected, or which can be plugged into matching sockets on a production device. The pinouts for these connectors are shown in Figure 1 below.

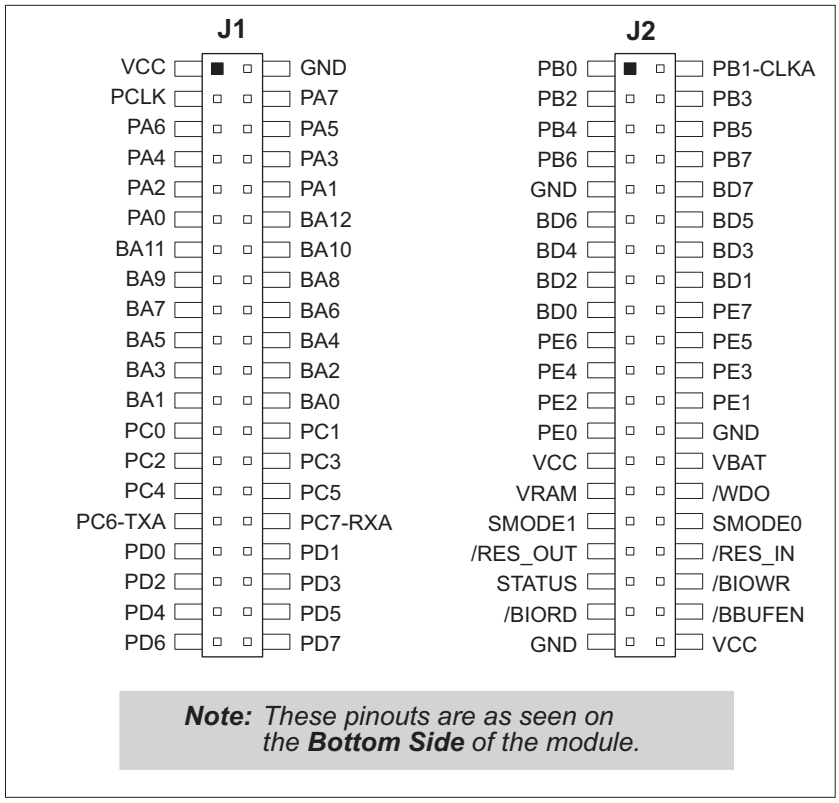



***Figure 1. RCM2100 Pinout***

## 1.2 Development Software

The RCM2100 modules use the Dynamic C development environment for rapid creation and debugging of runtime applications. Dynamic C provides a complete development environment with integrated editor, compiler and source-level debugger. It interfaces directly with the target system, eliminating the need for complex and unreliable in-circuit emulators.

Dynamic C must be installed on a Windows workstation with at least one free serial (COM) port for communication with the target system. See Chapter 3., "Software Installation & Overview," for complete information on installing Dynamic C.

> **NOTE:** The RCM2100 modules require Dynamic C v7.04 or later for development. A compatible version is included on the Development Kit CD-ROM.

## 1.3 How to Use This Manual

This ***Getting Started*** manual is intended to give users a quick but solid start with the RCM2100 modules. It does not contain detailed information on the module hardware capabilities, the Dynamic C development environment, or the TCP/IP software support for the integrated Ethernet port. Most users will want more detailed information on some or all of these topics in order to put the RCM2100 module to effective use.

### 1.3.1 Additional Product Information

Detailed information about the RCM2100 will be found in the ***RabbitCore RCM2100 User's Manual***, provided on the accompanying CD-ROM in both HTML and Adobe PDF format.

Some advanced users may choose to skip the rest of this introductory manual and proceed directly with the detailed hardware and software information in the User's Manual.

> **NOTE:** We recommend that anyone not thoroughly familiar with Z-World controllers at least read through the rest of this manual to gain the necessary familiarity to make use of the more advanced information.

### 1.3.2 Additional Reference Information

In addition to the product-specific information contained in the ***RabbitCore RCM2100 User's Manual***, several higher-level reference manuals are provided in HTML and PDF form on the accompanying CD-ROM. Advanced users will find these references valuable in developing systems based on the RCM2100 modules:

- ***Dynamic C User's Manual***
- ***An Introduction to TCP/IP***
- ***Dynamic C TCP/IP User's Manual***
- ***Rabbit 2000 Microprocessor User's Manual***

### 1.3.3 Using Online Documentation

We provide the bulk of our user and reference documentation in two electronic formats, HTML and Adobe PDF. We do this for several reasons.

We believe that providing all users with our complete library of product and reference manuals is a useful convenience. However, printed manuals are expensive to print, stock and ship. Rather than include and charge for manuals that every user may not want, or provide only product-specific manuals, we choose to provide our complete documentation and reference library in electronic form with every development kit and with our Dynamic C development environment.

> **NOTE:** The most current version of Adobe Acrobat Reader can always be downloaded from Adobe's web site at **http://www.adobe.com**.
> We recommend that you use version 4.0 or later.

Providing this documentation in electronic form saves an enormous amount of paper by not printing copies of manuals that users don't need. It reduces the number of outdated manuals we have to discard from stock as well, and it makes providing a complete library of manuals an almost cost-free option. For one-time or infrequent reference, electronic documents are more convenient than printed ones—after all, they aren't taking up shelf or desk space!

### Finding Online Documents

The online documentation is installed along with Dynamic C, and an icon for the documentation menu is placed on the workstation's desktop. Double-click this icon to reach the menu. If the icon is missing, create a new desktop icon that points to **default.htm** in the **docs** folder, found in the Dynamic C installation folder.

The latest versions of all documents are always available for free, unregistered download from our web sites as well.

### Printing Electronic Manuals

We recognize that many users prefer printed manuals for some uses. Users can easily print all or parts of those manuals provided in electronic form. The following guidelines may be helpful:

- Print from the Adobe PDF versions of the files, not the HTML versions.
- Print only the sections you will need to refer to more than once.
- Print manuals overnight, when appropriate, to keep from tying up shared resources during the work day.
- If your printer supports duplex printing, print pages double-sided to save paper and increase convenience.
- If you do not have a suitable printer or do not want to print the manual yourself, most retail copy shops (e.g. Kinkos, AlphaGraphics, etc.) will print the manual from the PDF file and bind it for a reasonable charge—about what we would have to charge for a printed and bound manual.

# 2. GETTING STARTED

This chapter describes the RCM2100 hardware in more detail, and explains how to set up and use the accompanying prototyping and development board.

**NOTE:** This chapter (and this manual) assume that you have the RabbitCore RCM2100 Development Kit. If you purchased an RCM2100 module by itself, you will have to adapt the information in this chapter and elsewhere to your test and development setup.

## 2.1 Development Kit Contents

The RCM2100 Development Kit contains the following items:

- RCM2100 module with Ethernet port, 512K flash memory and 512K SRAM.
- RCM2100 Prototyping Board with accessory hardware and components.
- Wall transformer power supply, 12 V DC, 1 A. (Included only with Development Kits sold for the North American market. Overseas users will have to substitute a power supply compatible with their local mains power.)
- 10-pin header to DE9 programming cable with integrated level-matching circuitry.
- ***Dynamic C*** CD-ROM, with complete product documentation on disk.
- This ***Getting Started*** manual.
- Registration card.

## 2.2 Overview of the Prototyping Board

The Prototyping Board included in the Development Kit makes it easy to connect an RCM2100 module to a power supply and a PC workstation for development. It also provides an array of basic I/O peripherals (switches and LEDs), as well as a prototyping area for more advanced hardware development.

For the most basic level of evaluation and development, the Prototyping Board can be used without modification.

As you progress to more sophisticated experimentation and hardware development, modifications and additions can be made to the board without modifying or damaging the RCM2100 module itself.

The Prototyping Board is shown in Figure 2 below, with its main features identified

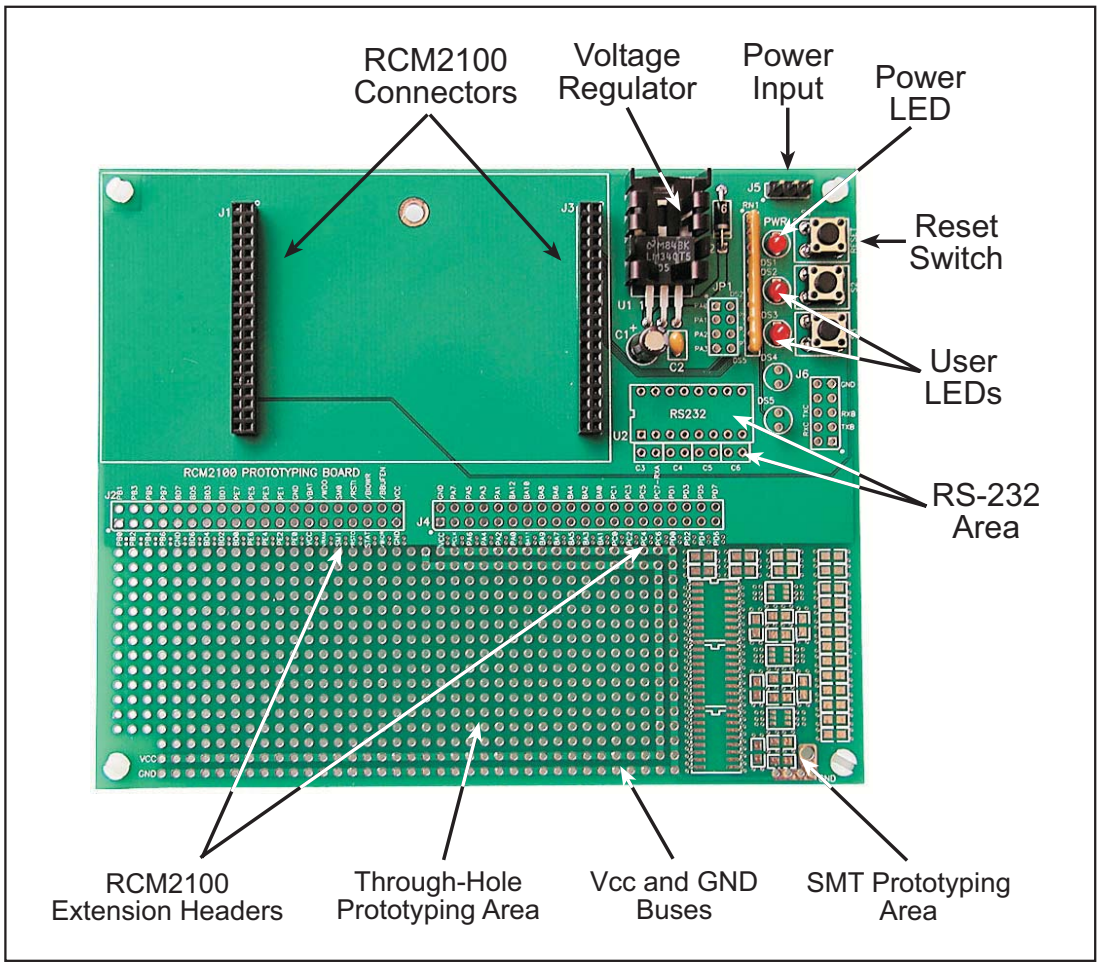



***Figure 2. RCM2100 Prototyping Board***

### 2.2.1 Prototyping Board Features

**Power Connection.** A 3-pin header is provided for connection of a power supply. Note that it is symmetrical, with both outer pins connected to ground and the center pin connected to the raw V+ input. The cable of the wall transformer provided with the North American version of the Development Kit ends in a connector that is correctly connected in either orientation.

Users providing their own power supply should ensure that it delivers 9–24 V DC at not less than 500 mA. The voltage regulator will get warm in use, but lower supply voltages will reduce thermal dissipation from the device.

**Regulated Power Supply.** The raw DC voltage provided at the POWER IN jack is routed to a 5 V linear voltage regulator, which provides stable power to the RCM2100 module and the Prototyping Board. A Shottky diode protects the power supply against damage from reversed raw power connections.

**Power LED.** The power LED lights whenever power is connected to the Prototyping Board.

**Reset Switch.** A momentary-contact, normally open switch is connected directly to the RCM2100's **/RES_IN** pin. Pressing the switch forces a hardware reset of the system.

**I/O Switches & LEDs.** Two momentary-contact, normally open switches are connected to the PB2 and PB3 pins of the RCM2100 module, and may be read as inputs by sample applications.

Two LEDs are connected to the PA0 and PA1 pins of the module, and may be driven as output indicators by sample applications. (Two more LEDs, driven by PA2 and PA3, may be added to the Prototyping Board for additional outputs.)

All the LEDs are connected through JP1, which has traces shorting adjacent pads together. These traces may be cut to disconnect the LEDs, and an 8-pin header soldered into JP1 to permit their selective reconnection with jumpers. See Figure 3 for details.

**Expansion Areas.** The Prototyping Board is provided with several unpopulated areas for expansion of I/O and interfacing capabilities. See the next section for details.

**Prototyping Area.** A generous prototyping area has been provided for the installation of through-hole components. Vcc (5 V DC) and Ground buses run around the edge of this area. An area for surface-mount devices is provided to the right of the through-hole area. (Note that there are SMT device pads on both top and bottom of the Prototyping Board.)

### 2.2.2 Prototyping Board Expansion

The Prototyping Board comes with several unpopulated areas, which may be filled with components to suit the user's development needs. After you have experimented with the sample programs in Chapter 4, you may wish to expand the board's capabilities for further experimentation and development. Refer to the Prototyping Board schematic (090–0116) for details as necessary.

**Module Extension Headers** The complete pin set of the RCM2100 module is duplicated at these two headers. Developers can solder wires directly into the appropriate holes, or, for more flexible development, two 40-pin header strips can be soldered into place. See Figure 1 on page 3 for the header pinouts.

**RS-232 Port** Two 2-wire or one 5-wire RS-232 serial port can be added to the Prototyping Board by installing a driver IC and four capacitors where indicated. The Maxim MAX232 driver chip or a similar device is recommended for U2. Refer to the Prototyping Board schematic for additional details.

A 10-pin 0.1" spacing header strip can be installed at J6 to permit connection of a ribbon cable leading to a standard DE-9 serial connector.

**NOTE:** The RS-232 chip, capacitors and header strip are available from electronics distributors such as Digi-Key and Mouser Electronics.

**Additional LEDs** Two additional LEDs (supplied with the development kit) can be soldered into place at DS4 and DS5. The cathode lead (longer of the two, marked by a flat on the LED case) should go towards the module.

**Prototyping Board Component Header** Several I/O pins from the module are hardwired to the Prototyping Board LEDs and switches.

To disconnect these devices and permit the pins to be used for other purposes, cut the traces between the pin rows. Use an exacto knife or similar tool to cut or break the traces crossing JP1, in the area indicated in Figure 3.

To permit selective reconnection of the devices, jumpers may be placed across the 8-pin header strip at JP1.

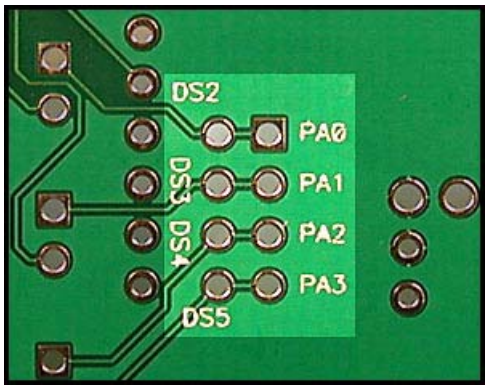



***Figure 3. Where to Cut Traces to Permanently Disable Demonstration Hardware on Prototyping Board***

## 2.3 Connections

There are three steps to connecting the Prototyping Board for use with Dynamic C and the sample programs:

1. Attach the RCM2100 module to the Prototyping Board.
2. Connect the programming cable between the RCM2100 module and the workstation PC.
3. Connect the power supply to the Prototyping Board.

### 2.3.1 Attach Module to Prototyping Board

Turn the RCM2100 module so that the Ethernet connector is on the left, as shown in Figure 4 below. Align the module headers J1 and J2 on the bottom side of the RCM2100 into header sockets J1 and J3 on the Prototyping Board.

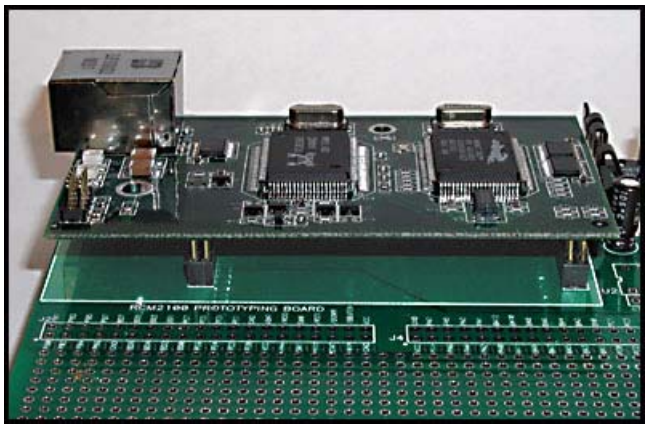


***Figure 4. Installing the RCM2100 Module on the Prototyping Board. Note the orientation of the module.***

**NOTE:** It is important that you line up the RCM2100 pins on headers J1 and J2 exactly with the corresponding pins of header sockets J1 and J3 on the Prototyping Board. The header pins may become bent or damaged if the pin alignment is offset, and the module will not work.

Press the module's pins firmly into the Prototyping Board headers. The installed module is shown in Figure 5 below.

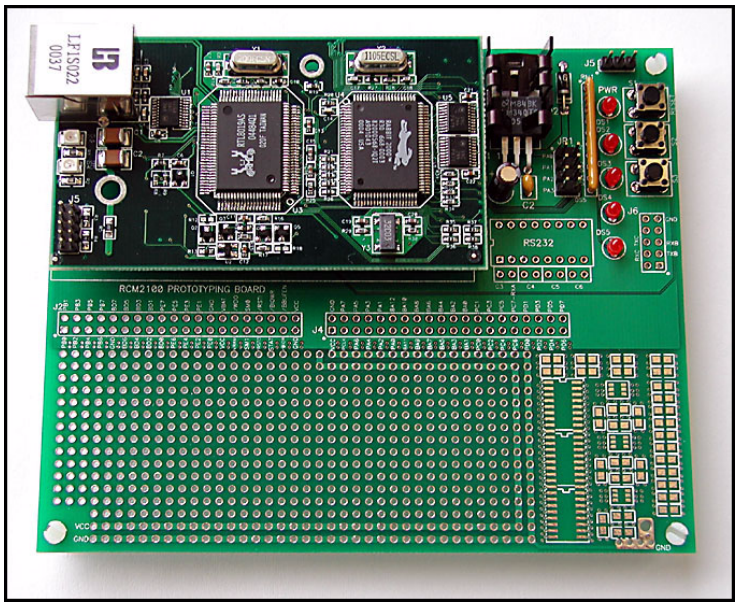



***Figure 5. RCM2100 Installed and Seated on the Prototyping Board***

### 2.3.2 Connect Programming Cable

The programming cable connects the RCM2100 module to the PC running Dynamic C, to download programs and to monitor the RCM2100 for debugging.

Connect the 10-pin connector of the programming cable labeled **PROG** to header J5 on the RCM2100 module as shown in Figure 6 below. Be sure to orient the red edge of the cable towards pin 1 of the connector. (Do not use the **DIAG** connector, which is used for a normal serial connection.)

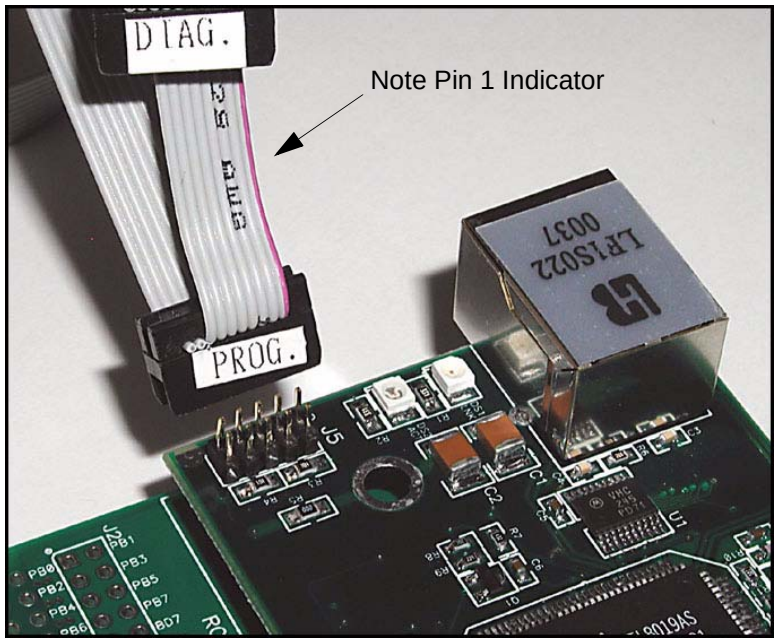



***Figure 6. Attaching Programming Cable to the RCM2100***

**NOTE:** The stripe on the cable is towards pin 1 of the header J5.

Connect the other end of the programming cable to a COM port on your PC. Make a note of the port to which you connect the cable, as Dynamic C needs to have this parameter configured when it is installed.

**NOTE:** COM 1 is the default port used by Dynamic C.

**NOTE:** Some PCs now come equipped only with a USB port. It may be possible to use an RS-232/USB converter with the programming cable supplied with your RCM2100 module. An RS-232/USB converter is available through the Z-World Web store.

### 2.3.3 Connect Power

When all other connections have been made, you can connect power to the RCM2100 Prototyping Board.

Hook the connector from the wall transformer to header J5 on the Prototyping Board as shown in Figure 7 below. The connector may be attached either way as long as it is not offset to one side.

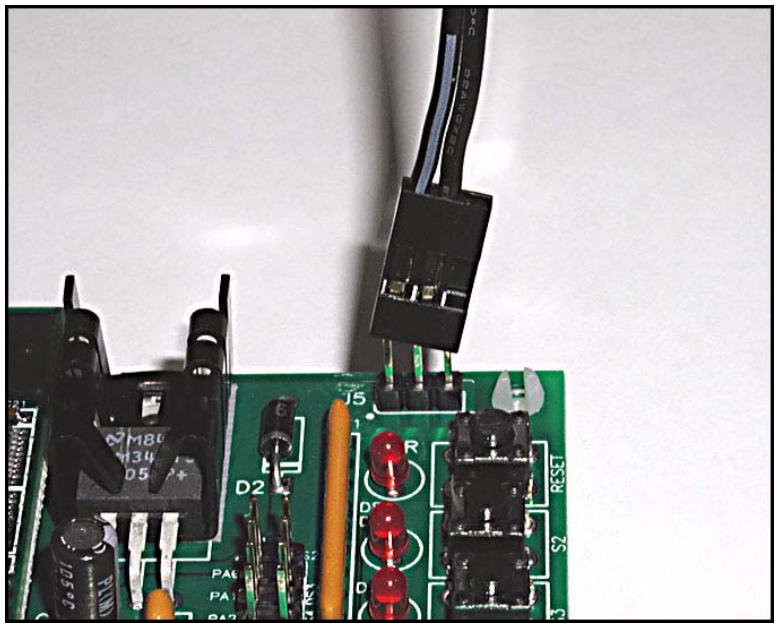



***Figure 7. Power Supply Connections to Prototyping Board***

Plug in the wall transformer. The power LED on the Prototyping Board should light up. The RCM2100 and the Prototyping Board are now ready to be used.

> **NOTE:** A **RESET** button is provided on the Prototyping Board to allow hardware reset without disconnecting power.

To power down the Prototyping Board, unplug the power connector from J5. You should disconnect power before making any circuit adjustments in the prototyping area, changing any connections to the board, or removing the RCM2100 module from the Prototyping Board.

## 2.4 Run a Sample Program

If you already have Dynamic C installed, you are now ready to test your programming connections by running a sample program.

If you are using a USB port to connect your computer to the RCM2100 module, choose **Options > Project Options** and select "Use USB to Serial Converter" under the **Communications** tab.

Find the file **PONG.C**, which is in the Dynamic C **SAMPLES** folder. To run the program, open it with the **File** menu (if it is not still open), then compile and run it by pressing **F9** or by selecting **Run** in the **Run** menu. The **STDIO** window will open and will display a small square bouncing around in a box.

### 2.4.1 Troubleshooting

If Dynamic C appears to compile the BIOS successfully, but you then receive a communication error message when you compile and load the sample program, it is possible that your PC cannot handle the higher program-loading baud rate. Try changing the maximum download rate to a slower baud rate as follows.

- Locate the **Serial Options** dialog in the Dynamic C **Options > Project Options > Communications** menu. Select a slower Max download baud rate.

If a program compiles and loads, but then loses target communication before you can begin debugging, it is possible that your PC cannot handle the default debugging baud rate. Try lowering the debugging baud rate as follows.

- Locate the **Serial Options** dialog in the Dynamic C **Options > Project Options > Communications** menu. Choose a lower debug baud rate.

If there are any other problems:

- Check to make sure you are using the **PROG** connector, not the **DIAG** connector, on the programming cable.
- Check both ends of the programming cable to ensure that they are firmly plugged into the PC and the programming port on the RCM2100.
- Ensure that the RCM2100 module is firmly and correctly installed in its connectors on the Prototyping Board.
- Select a different COM port within Dynamic C. From the **Options** menu, select **Project Options**, then select **Communications**. Select another COM port from the list, then click OK. Press **<Ctrl-Y>** to force Dynamic C to recompile the BIOS. If Dynamic C still reports it is unable to locate the target system, repeat the above steps until you locate the active COM port.

## 2.5 Where Do I Go From Here?

If everything appears to be working, we recommend the following sequence of action:

1. Run all of the sample programs described in Chapter 3 to get a basic familiarity with Dynamic C and the RabbitCore module's capabilities.
2. For further development, refer to the ***RabbitCore RCM2100 User's Manual*** for details of the module's hardware and software components.

   A documentation icon should have been installed on your workstation's desktop; click on it to reach the documentation menu. You can create a new desktop icon that points to **default.htm** in the **docs** folder in the Dynamic C installation folder.
3. For advanced development topics, refer to the ***Dynamic C User's Manual*** and the ***Dynamic C TCP/IP User's Manual***, also in the online documentation set.

### 2.5.1 Technical Support

**NOTE:** If you purchased your RCM2100 through a distributor or through a Z-World or Rabbit Semiconductor partner, contact the distributor or partner first for technical support.

If there are any problems at this point:

- Use the Dynamic C **Help** menu to get further assistance with Dynamic C.
- Check the Z-World/Rabbit Semiconductor Technical Bulletin Board at www.zworld.com/support/bb/.
- Use the Technical Support e-mail form at www.zworld.com/support/questionSubmit.shtml.

# 3. SOFTWARE INSTALLATION & OVERVIEW

To develop and debug programs for the RCM2100 (and for all other Z-World and Rabbit Semiconductor hardware), you must install and use Dynamic C. Dynamic C is an integrated development system for writing embedded software. It runs on an IBM-compatible PC and is designed for use with Z-World single-board computers and other single-board computers based on the Rabbit microprocessor. This chapter takes you through the installation of Dynamic C, and then provides a tour of the sample programs for the RCM2100.

## 3.1 An Overview of Dynamic C

Dynamic C has been in use worldwide since 1989. It is specially designed for programming embedded systems, and features quick compile and interactive debugging. A complete reference guide to Dynamic C is contained in the ***Dynamic C User's Manual***.

You have a choice of doing your software development in the flash memory or in the SRAM included on the RCM2100. The flash memory and SRAM options are selected with the **Options > Project Options > Compiler** menu.

The advantage of working in RAM is to save wear on the flash memory, which is limited to about 100,000 write cycles. The disadvantage is that the code and data might not both fit in RAM.

**NOTE:** An application can be developed in RAM, but cannot run standalone from RAM after the programming cable is disconnected. All standalone applications can only run from flash memory.

**NOTE:** Do not depend on the flash memory sector size or type. Due to the volatility of the flash memory market, the RCM2000 and Dynamic C were designed to accommodate flash devices with various sector sizes.

Developing software with Dynamic C is simple. Users can write, compile, and test C and assembly code without leaving the Dynamic C development environment. Debugging occurs while the application runs on the target. Alternatively, users can compile a program to an image file for later loading. Dynamic C runs on PCs under Windows 95, 98, 2000, NT, Me, and XP. Programs can be downloaded at baud rates of up to 460,800 bps after the program compiles.

Dynamic C has a number of standard features.

- Full-feature source and/or assembly-level debugger, no in-circuit emulator required.
- Royalty-free TCP/IP stack with source code and most common protocols.
- Hundreds of functions in source-code libraries and sample programs:
  - Exceptionally fast support for floating-point arithmetic and transcendental functions.
  - RS-232 and RS-485 serial communication.
  - Analog and digital I/O drivers.
  - $I^2C$, SPI, GPS, file system.
  - LCD display and keypad drivers.
- Powerful language extensions for cooperative or preemptive multitasking
- Loader utility program to load binary images into Z-World targets in the absence of Dynamic C.
- Provision for customers to create their own source code libraries and augment on-line help by creating "function description" block comments using a special format for library functions.
- Standard debugging features:
  - Breakpoints—Set breakpoints that can disable interrupts.
  - Single-stepping—Step into or over functions at a source or machine code level, µC/OS-II aware.
  - Code disassembly—The disassembly window displays addresses, opcodes, mnemonics, and machine cycle times. Switch between debugging at machine-code level and source-code level by simply opening or closing the disassembly window.
  - Watch expressions—Watch expressions are compiled when defined, so complex expressions including function calls may be placed into watch expressions. Watch expressions can be updated with or without stopping program execution.
  - Register window—All processor registers and flags are displayed. The contents of general registers may be modified in the window by the user.
  - Stack window—shows the contents of the top of the stack.
  - Hex memory dump—displays the contents of memory at any address.
  - **STDIO** window—`printf` outputs to this window and keyboard input on the host PC can be detected for debugging purposes. `printf` output may also be sent to a serial port or file.

## 3.2 Installing Dynamic C

Insert the Dynamic C CD from the Development Kit in your PC's CD-ROM drive. If the installation does not auto-start, run the **setup.exe** program in the root directory of the Dynamic C CD. Install any Dynamic C modules after you install Dynamic C.

Dynamic C has two components that can be installed together or separately. One component is Dynamic C itself, with the development environment, support files and libraries. The other component is the documentation library in HTML and PDF formats, which may be left uninstalled to save hard drive space or installed elsewhere (on a separate or network drive, for example).

The installation type is selected in the installation menu. The options are:

- **Typical Installation —** Both Dynamic C and the documentation library will be installed in the specified folder (default).
- **Compact Installation —** Only Dynamic C will be installed.
- **Custom Installation —** You will be allowed to choose which components are installed. This choice is useful to install or reinstall just the documentation.

### 3.2.1 Early Versions of Dynamic C

If you are using Dynamic C version 7.04 or earlier, modify the BIOS source code as follows. Skip these three steps if your version of Dynamic C is 7.05 or later.

1. Open the BIOS source code file named **RABBITBIOS.C**, which can be found in the **BIOS** directory.
2. Change the line

   ```
   #define USE115KBAUD 1   // set to 0 to use 57600 baud
   ```

   to read as follows.

   ```
   #define USE115KBAUD 0   // set to 0 to use 57600 baud
   ```

3. Save the changes using **File > Save**.

Now press **<Ctrl-Y>**. You should receive the **"BIOS successfully compiled …"** message indicating that the target is now ready to compile a program.

## 3.3 Sample Programs

To help familiarize you with the RCM2100 modules, several sample Dynamic C programs have been included. Loading, executing and studying these programs will give you a solid hands-on overview of the RCM2100's capabilities, as well as a quick start with Dynamic C as an application development tool. These programs are intended to serve as tutorials, but then can also be used as starting points or building blocks for your own applications.

> **NOTE:** It is assumed in this section that you have at least an elementary grasp of ANSI C. If you do not, see the introductory pages of the ***Dynamic C User's Manual*** for a suggested reading list.

Each sample program has comments that describe the purpose and function of the program.

Before running any of these sample programs, make sure that your RCM2100 is connected to the Prototyping Board and to your PC as described in Section 2.3, "Connections."

To run a sample program, open it with the **File** menu (if it is not already open), then compile and run it by pressing **F9** or by selecting **Run** in the **Run** menu.

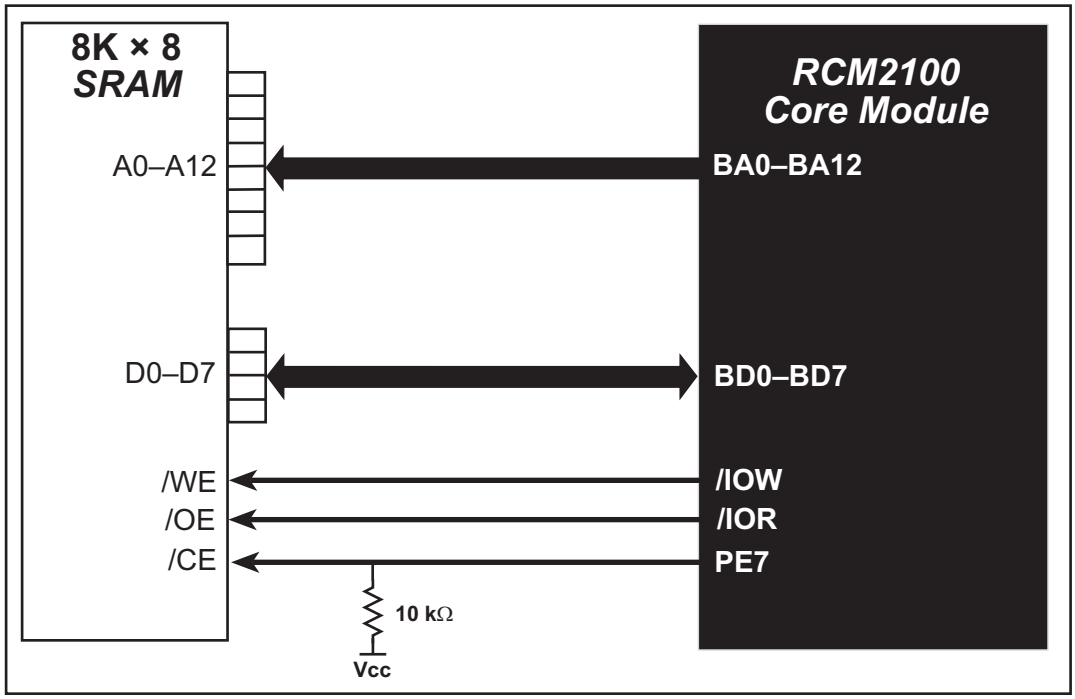
More complete information on Dynamic C is provided in the ***Dynamic C User's Manual***.

### 3.3.1 Getting to Know the RCM2100

The following sample programs can be found in the **SAMPLES\RCM2100** folder.

- **EXTSRAM.C**—demonstrates the setup and simple addressing to an external SRAM. This program first maps the external SRAM to the I/O Bank 7 register with a maximum of 15 wait states, chip select strobe (PE7), and allows writes. The first 256 bytes of SRAM are cleared and read back. Values are then written to the same area and are read back. The Dynamic C **STDIO** window will indicate if writes and reads did not occur

  Connect an external SRAM as shown below before you run this sample program.




- **FLASHLED.C**—repeatedly flashes LED DS3 on the Prototyping Board on and off. LED DS3 is controlled by Parallel Port A bit 1 (PA1).
- **FLASHLED2.C**—repeatedly flashes LED DS3 on the Prototyping Board on and off. LED DS3 is controlled by Parallel Port A bit 1 (PA1).

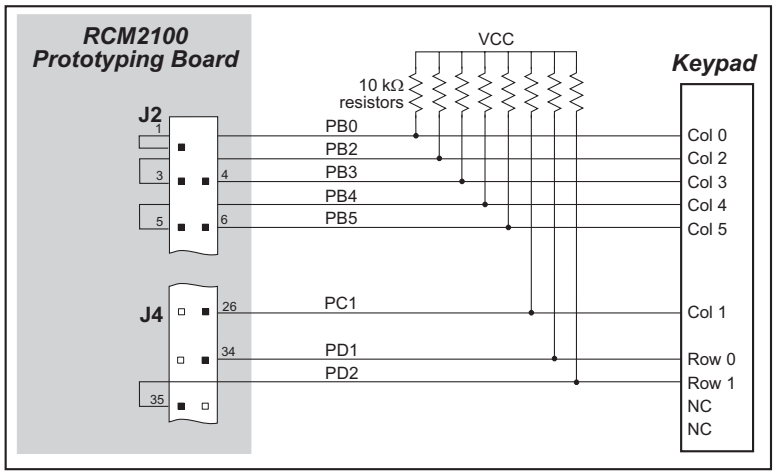
  This sample program also shows the use of the **runwatch()** function to allow Dynamic C to update watch expressions while running. The following steps explain how to do this.

  1. Add a watch expression for "k" in the **Inspect > Add Watch** dialog box.
  2. Click "Add" or "Add to top" so that it will be in the watch list permanently.
  3. Click **OK** to close the dialog box.
  4. Press **<Ctrl+U>** while the program is running. This will update the watch window.

- **FLASHLEDS.C**—demonstrates the use of coding with assembly instructions, cofunctions, and costatements to flash LEDs DS2 and DS3 on the Prototyping Board on and off. LEDs DS2 and DS3 are controlled by Parallel Port A bit 0 (PA0) and Parallel Port A bit 1 (PA1).Once you have compile this program and it is running, LEDs DS2 and DS3 will flash on/off at different rates.
- **FLASHLEDS2.C**—demonstrates the use of cofunctions and costatements to flash LEDs DS2 and DS3 on the Prototyping Board on and off. LEDs DS2 and DS3 are controlled by Parallel Port A bit 0 (PA0) and Parallel Port A bit 1 (PA1).Once you have compile this program and it is running, LEDs DS2 and DS3 will flash on/off at different rates.
- **KEYLCD2.C**—demonstrates a simple setup for a 2 × 6 keypad and a 2 × 20 LCD.

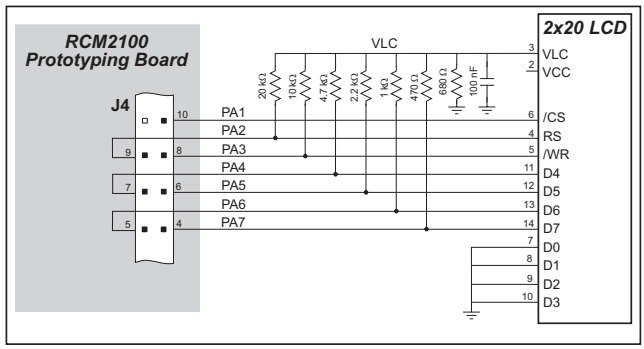
  Connect the keypad to Parallel Ports B, C, and D.

  PB0—Keypad Col 0
  PC1—Keypad Col 1
  PB2—Keypad Col 2
  PB3—Keypad Col 3
  PB4—Keypad Col 4
  PB5—Keypad Col 5
  PD1—Keypad Row 0
  PD2—Keypad Row 1




  Connect the LCD to Parallel Port A.

  )PA0—backlight (if connected
  PA1—LCD /CS
  PA2—LCD RS (High = Control, Low = Data) / LCD Contrast 0
  PA3—LCD /WR/ LCD Contrast 1
  PA4—LCD D4 / LCD Contrast 2
  PA5—LCD D5 / LCD Contrast 3
  PA6—LCD D6 / LCD Contrast 4
  PA7—LCD D7 / LCD Contrast 5

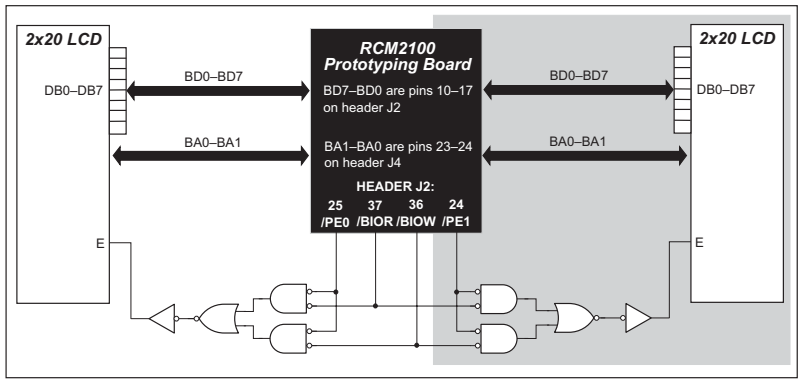



  Once the connections have been made and the sample program isrunning, the LCD will display two rows of 6 dots, each dot representing the corresponding key. When a key is pressed, the corresponding dot will become an asterisk.

- **LCD_DEMO.C**—demonstrates a simple setup for an LCD that uses the HD44780 controller or an equivalent.

  Connect the LCD to the RCM2100 address and data lines on the Prototyping Board.

  BD0—DB0
  BD1—DB1
  BD2—DB2
  BD3—DB3
  BD4—DB4
  BD5—DB5
  BD6—DB6
  BD7—DB7




  BA0—RS (Register Select: 0 = command, 1 = data)
  BA1—R/W (0=write, 1=read)
  *—E (normally low: latches on high-to-low transition)

- **SWTEST.C**—demonstrates the use of pushbutton switches S2 and S3 to toggle LEDs DS2 and DS3 on the Prototyping Board on and off.

  Parallel Port A bit 0 = LED DS2
  Parallel Port A bit 1 = LED DS3

  Parallel Port B bit 2 = switch S2
  Parallel Port B bit 3 = switch S3

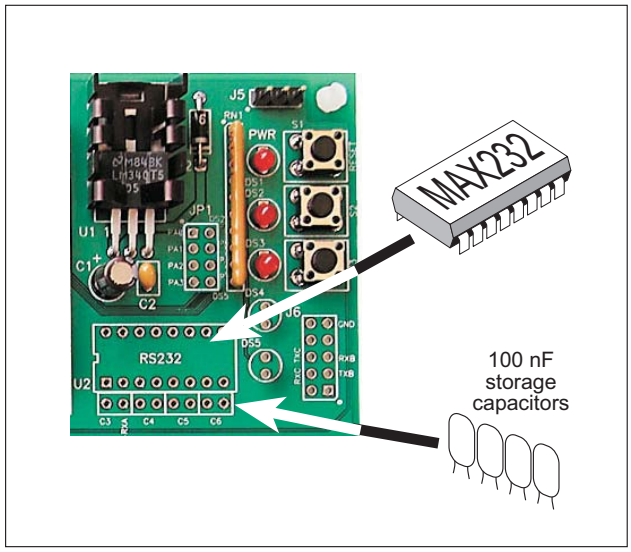
- **TOGGLELED.C**—demonstrates the use of costatements to detect switch presses using the press-and-release method of debouncing. As soon as the sample program starts running, LED DS3 on the Prototyping Board (which is controlled by PA1) starts flashing once per second. Press switch S2 on the Prototyping Board (which is connected to PB2) to toggle LED DS2 on the Prototyping Board (which is controlled by PA0). The pushbutton switch is debounced by the software.

### 3.3.2 Serial Communication

The following sample programs can be found in the **SAMPLES\RCM2100** folder.

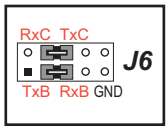
Two sample programs, **CORE_FLOWCONTROL.C** and **CORE_PARITY.C**, are available to illustrate RS-232 communication. To run these sample programs, you will have to add an RS-232 transceiver such as the MAX232 at location U2 and four 100 nF capacitors at C3–C6 on the Prototyping Board. Also install the 2 × 5 IDC header included with the Prototyping Board accessory parts at J6 to interface the RS-232 signals.

The diagram shows the connections.




- **CORE_FLOWCONTROL.C**—This program demonstrates hardware flow control by configuring Serial Port C (PC3/PC2) for CTS/RTS with serial data coming from TxB at 115,200 bps. One character at a time is received and is displayed in the **STDIO** window.

  To set up the Prototyping Board, you will need to tie PC4 and PC5 (TxB and RxB) together at header J4, and you will also tie PC2 and PC3 (TxC and RxC) together using the jumpers supplied in the Development Kit as shown in the diagram.

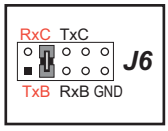



  A repeating triangular pattern should print out in the **STDIO** window. The program will periodically switch flow control on or off to demonstrate the effect of no flow control.

  Refer to the **serBflowcontrolOn()** function call in the ***Dynamic C Function Reference Manual*** for a general description on how to set up flow control lines.

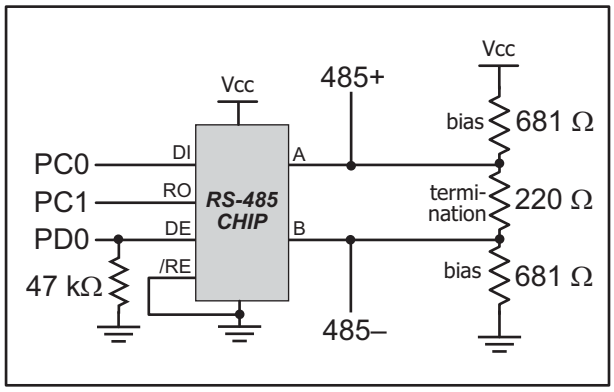
- **CORE_PARITY.C**—This program demonstrates the use of parity modes by repeatedly sending byte values 0–127 from Serial Port B to Serial Port C. The program will switch between generating parity or not on Serial Port B. Serial Port C will always be checking parity, so parity errors should occur during every other sequence.

  To set up the Prototyping Board, you will need to tie PC4 and PC3 (TxB and RxC) together at header J4 using the jumpers supplied in the Development Kit as shown in the diagram.




  The Dynamic C **STDIO** window will display the error sequence.

Two sample programs, **MASTER2.C** and **SLAVE2.C**, are available to illustrate RS-485 master/slave communication. To run these sample programs, you will need a second Rabbit-based system with RS-485, and you will also have to add an RS-485 transceiver such as the SP483E and bias resistors to the Prototyping Board.




The diagram shows the connections. You will have to connect PC0 and PC1 (Serial Port D) on the Prototyping Board to the RS-485 transceiver, and you will connect PD0 to the RS-485 transceiver to enable or disable the RS-485 transmitter.

The RS-485 connections between the slave and master devices are as follows.

- RS485+ to RS485+
- RS485– to RS485–
- GND to GND

- **MASTER2.C**—This program demonstrates a simple RS-485 transmission of lower case letters to a slave RCM2100. The slave will send back converted upper case letters back to the master RCM2100 and display them in the **STDIO** window. Use **SLAVE2.C** to program the slave RCM2100.
- **SLAVE2.C**—This program demonstrates a simple RS-485 transmission of lower case letters to a master RCM2100. The slave will send back converted upper case letters back to the master RCM2100 and display them in the **STDIO** window. Use **MASTER2.C** to program the master RCM2100.

### 3.3.3 Other Sample Programs

Section 4.7 covers how to run the TCP/IP sample programs, which are then described in detail.

### 3.3.4 Sample Program Descriptions

#### 3.3.4.1 FLASHLED.C

This program is about as simple as a Dynamic C application can get—the equivalent of the traditional "Hello, world!" program found in most basic programming tutorials. If you are familiar with ANSI C, you should have no trouble reading through the source code and understanding it.

The only new element in this sample application should be Dynamic C's handling of the Rabbit microprocessor's parallel ports. The program:

4. Initializes the pins of Port A as outputs.
5. Sets all of the pins of Port A high, turning off the attached LEDs.
6. Starts an endless loop with a **`for(;;)`** expression, and within that loop:
    - Writes a bit to turn bit 1 off, lighting LED DS3;
    - Waits through a delay loop;
    - Writes a bit to turn bit 1 on, turning off the LED;
    - Waits through a second delay loop;

    These steps repeat as long as the program is allowed to run.

You can change the flash rate of the LED by adjusting the loop values in the two **`for`** expressions. The first loop controls the LED's "off" time; the second loop controls its "on" time.

> **NOTE:** Since the variable **`j`** is defined as type **`int`**, the range for **`j`** must be between 0 and 32767. To permit larger values and thus longer delays, change the declaration of **`j`** to **`unsigned int`** or **`long`**.

#### More Information

See the section on primitive data types, and the entries for the library functions **`WrPortI( )`** and **`BitWrPortI( )`** in the ***Dynamic C User's Manual***.

#### 3.3.4.2 FLASHLEDS.C

In addition to Dynamic C's implementation of C-language programming for embedded systems, it supports assembly-language programming for very efficient processor-level control of the module hardware and program flow. This application is similar to **FLASHLED.C** and **TOGGLELEDS.C**, but uses assembly language for the low-level port control within *cofunctions*, another powerful multitasking tool.

Dynamic C permits the use of assembly language statements within C code. This program creates three functions using assembly language statements, then creates a C cofunction to call two of them. That cofunction is then called within **main()**.

Within each of the C-like functions, the **#asm** and **#endasm** directives are used to indicate the beginning and end of the assembly language statements.

In the function **initialize_ports( )**, port A is initialized to be all outputs while bit 0 of port E is initialized to be an output.

In the function **ledon()**, a 0 is written to the port A bit corresponding to the desired LED (0, which equals DS3, or 1 which equals DS4), turning that LED on. The **ledoff( )** function works exactly the same way except that a 1 is written to the bit, turning the selected LED off.

Finally, in the cofunction **flashled()**, the LED to be flashed, the on time in milliseconds, and the off time in milliseconds are passed as arguments. This function uses an endless **for(;;)** loop to call the **ledon()** and **ledoff()** functions, separated by calls to the wait function **DelayMs()**. This sequence will make the indicated LED flash on and off.

As is proper in C program design, the contents of **main()** are almost trivial. The program first calls **initialize_ports()**, then begins an endless **for(;;)** loop. Within this loop, the program:

1. Calls the library function **hitwd()**, which resets the microprocessor's watchdog timer. (If the watchdog timer is not reset every so often, it will force a hard reset of the system. The purpose is to keep an intermittent program or hardware fault from locking up the system. Normally, this function is taken care of by the virtual driver, but it is called explicitly here).
2. Sets up a costatement which calls two instances of the **flashled()** function, one for each LED. Note that one LED is flashed one second on, one-half second (500 ms) off, while the other is flashed in the reverse pattern.

Note also the **wfd** keyword in the costatement. This keyword (an abbreviation for **waitfordone**, which can also be used) must be used when calling cofunctions. For a complete explanation, see Section 5 and 6 in the ***Dynamic C User's Manual***.

### More Information

See the entries for the **hitwd()** and **DelayMs()** functions in the ***Dynamic C User's Manual***, as well as those for the directives **#asm** and **#endasm**. For a complete explana-

tion of how Dynamic C handles multitasking with costatements and cofunctions, see Chapter 5, "Multitasking with Dynamic C," and Chapter 6, "The Virtual Driver," in the ***Dynamic C User's Manual***.

#### 3.3.4.3 TOGGLELED.C

One of Dynamic C's unique and powerful aspects is its ability to efficiently multitask using *cofunctions* and *costatements*. This simple application demonstrates how these program elements work.

This sample program uses two costatements to set up and manage the two tasks. Costatements must be contained in a loop that will "tap" each of them at regular intervals. This program:

1. Initializes the pins of Port A as outputs.
2. Sets all the pins of Port A high, turning off the attached LEDs.
3. Sets the toggled LED status variable **`vswitch`** to 0 (LED off).
4. Starts an endless loop using a **`while(1)`** expression, and within that loop:
    - Executes a costatement that flashes LED DS3;
    - Executes a costatement that checks the state of switch S2 and toggles the state of **`vswitch`** if it is pressed;
    - Turns LED DS2 on or off, according to the state of **`vswitch`**.

    These steps repeat as long as the program is allowed to run.

The first costatement is a compressed version of **`FLASHLED.c`**, with slightly different flash timing. It also uses the library function **`DelayMs()`** to deliver more accurate timing than the simple delay loops of the previous program.

The second costatement does more than check the status of S2. Switch contacts often "bounce" open and closed several times when the switch is actuated, and each bounce can be interpreted by fast digital logic as an independent press. To clean up this input, the code in the second costatement "debounces" the switch signal by waiting 50 milliseconds and checking the state of the switch again. If it is detected as being closed both times, the program considers it a valid switch press and toggles **`vswitch`**.

Unlike most C statements, the two costatements are not executed in their entirety on each iteration of the **`while(1)`** loop. Instead, the list of statements within each costatement is initiated on the first loop, and then executed one "slice" at a time on each successive interation. This mode of operation is known as a *state machine*, a powerful concept that permits a single processor to efficiently handle a number of independent tasks.

The ability of Dynamic C to manage state machine programs enables you to create very powerful and efficient embedded systems with much greater ease than other programming methods.

### More Information

See the entries for the **`DelayMs()`** function, as well as Section 5, "Multitasking with Dynamic C," in the ***Dynamic C User's Manual***.

## 3.4 Upgrading Dynamic C

Dynamic C patches that focus on bug fixes are available from time to time. Check the Web sites

- www.zworld.com/support/

or

- www.rabbitsemiconductor.com/support/

for the latest patches, workarounds, and bug fixes.

### 3.4.1 Add-On Modules

Dynamic C installations are designed for use with the board they are included with, and are included at no charge as part of our low-cost kits. Z-World offers add-on Dynamic C modules for purchase, including the popular μC/OS-II real-time operating system, as well as PPP, Advanced Encryption Standard (AES), and other select libraries.

In addition to the Web-based technical support included at no extra charge, a one-year telephone-based technical support module is also available for purchase.

# 4. USING THE TCP/IP FEATURES

## 4.1 TCP/IP Connections

Programming and development can be done with the RCM2100 RabbitCore modules without connecting the Ethernet port to a network. However, if you will be running the sample programs that use the Ethernet capability or will be doing Ethernet-enabled development, you should connect the RCM2100 module's Ethernet port at this time.

Before proceeding you will need to have the following items.

- If you don't have Ethernet access, you will need at least a 10Base-T Ethernet card (available from your favorite computer supplier) installed in a PC.
- Two RJ-45 straight through Ethernet cables and a hub, or an RJ-45 crossover Ethernet cable.

The Ethernet cables and a 10Base-T Ethernet hub are available from Z-World in a TCP/IP tool kit. More information is available at www.zworld.com.

1. Connect the AC adapter and the programming cable as shown in Chapter 2, "Getting Started."
2. Ethernet Connections

   There are four options for connecting the RCM2100 module to a network for development and runtime purposes. The first two options permit total freedom of action in selecting network addresses and use of the "network," as no action can interfere with other users. We recommend one of these options for initial development.

- **No LAN —** The simplest alternative for desktop development. Connect the RCM2100's Ethernet port directly to the PC's network interface card using an RJ-45 *crossover cable*. A crossover cable is a special cable that flips some connections between the two connectors and permits direct connection of two client systems. A standard RJ-45 network cable will not work for this purpose.
- **Micro-LAN —** Another simple alternative for desktop development. Use a small Ethernet 10Base-T hub and connect both the PC's network interface card and the RCM2100's Ethernet port to it, using standard network cables.

The following options require more care in address selection and testing actions, as conflicts with other users, servers and systems can occur:

- **LAN —** Connect the RCM2100's Ethernet port to an existing LAN, preferably one to which the development PC is already connected. You will need to obtain IP addressing information from your network administrator.
- **WAN —** The RCM2100 is capable of direct connection to the Internet and other Wide Area Networks, but exceptional care should be used with IP address settings and all network-related programming and development. We recommend that development and debugging be done on a local network before connecting a RabbitCore system to the Internet.

  **TIP:** Checking and debugging the initial setup on a micro-LAN is recommended before connecting the system to a LAN or WAN.

The PC running Dynamic C through the serial port on the RCM2100 does not need to be the PC with the Ethernet card.

3. Apply Power

   Plug in the AC adapter. The RCM2100 module is now ready to be used.

## 4.2 TCP/IP Primer on IP Addresses

Obtaining IP addresses to interact over an existing, operating, network can involve a number of complications, and must usually be done with cooperation from your ISP and/or network systems administrator. For this reason, it is suggested that the user begin instead by using a direct connection between a PC and the RCM2100 board using an Ethernet crossover cable or a simple arrangement with a hub. (A crossover cable should not be confused with regular straight through cables.)

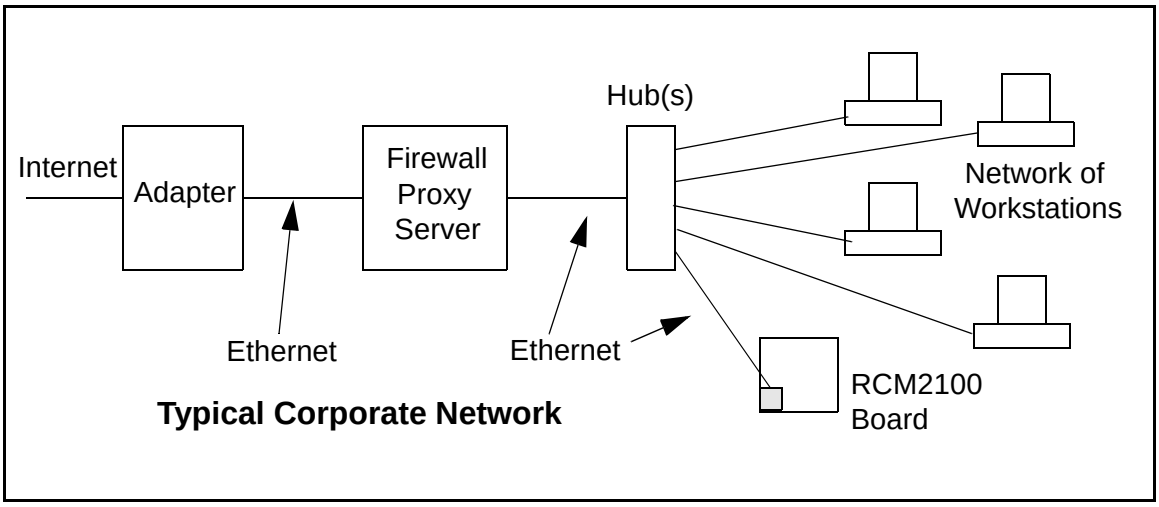
In order to set up this direct connection, the user will have to use a PC without networking, or disconnect a PC from the corporate network, or install a second Ethernet adapter and set up a separate private network attached to the second Ethernet adapter. Disconnecting your PC from the corporate network may be easy or nearly impossible, depending on how it is set up. If your PC boots from the network or is dependent on the network for some or all of its disks, then it probably should not be disconnected. If a second Ethernet adapter is used, be aware that Windows TCP/IP will send messages to one adapter or the other, depending on the IP address and the binding order in Microsoft products. Thus you should have different ranges of IP addresses on your private network from those used on the corporate network. If both networks service the same IP address, then Windows may send a packet intended for your private network to the corporate network. A similar situation will take place if you use a dial-up line to send a packet to the Internet. Windows may try to send it via the local Ethernet network if it is also valid for that network.

The following IP addresses are set aside for local networks and are not allowed on the Internet:  10.0.0.0 to 10.255.255.255, 172.16.0.0 to 172.31.255.255, and 192.168.0.0 to 192.168.255.255.

The RCM2100 board uses a 10Base-T type of Ethernet connection, which is the most common scheme. The RJ-45 connectors are similar to U.S. style telephone connectors, except they are larger and have 8 contacts.

An alternative to the direct connection using a crossover cable is a direct connection using a hub. The hub relays packets received on any port to all of the ports on the hub. Hubs are low in cost and are readily available. The RCM2100 board uses 10 Mbps Ethernet, so the hub or Ethernet adapter must be either a 10 Mbps unit or a 10/100 unit that adapts to either 10 or 100 Mbps.

In a corporate setting where the Internet is brought in via a high-speed line, there are typically machines between the outside Internet and the internal network. These machines include a combination of proxy servers and firewalls that filter and multiplex Internet traffic. In the configuration below, the RCM2100 board could be given a fixed address so any of the computers on the local network would be able to contact it. It may be possible to configure the firewall or proxy server to allow hosts on the Internet to directly contact the controller, but it would probably be easier to place the controller directly on the external network outside of the firewall. This avoids some of the configuration complications by sacrificing some security.




If your system administrator can give you an Ethernet cable along with its IP address, the netmask and the gateway address, then you may be able to run the sample programs without having to set up a direct connection between your computer and the RCM2100 board. You will also need the IP address of the nameserver, the name or IP address of your mail server, and your domain name for some of the sample programs.

## 4.3 IP Addresses Explained

IP (Internet Protocol) addresses are expressed as 4 decimal numbers separated by periods, for example:

216.103.126.155

10.1.1.6

Each decimal number must be between 0 and 255. The total IP address is a 32-bit number consisting of the 4 bytes expressed as shown above. A local network uses a group of adjacent IP addresses. There are always $2^N$ IP addresses in a local network. The netmask (also called subnet mask) determines how many IP addresses belong to the local network. The netmask is also a 32-bit address expressed in the same form as the IP address. An example netmask is:

255.255.255.0

This netmask has 8 zero bits in the least significant portion, and this means that $2^8$ addresses are a part of the local network. Applied to the IP address above (216.103.126.155), this netmask would indicate that the following IP addresses belong to the local network:

216.103.126.0

216.103.126.1

216.103.126.2

etc.

216.103.126.254

216.103.126.255

The lowest and highest address are reserved for special purposes. The lowest address (216.103.126.0) is used to identify the local network. The highest address (216.103.126.255) is used as a broadcast address. Usually one other address is used for the address of the gateway out of the network. This leaves 256 - 3 = 253 available IP addresses for the example given.

## 4.4 How IP Addresses are Used

The actual hardware connection via an Ethernet uses Ethernet adapter addresses (also called MAC addresses). These are 48-bit addresses and are unique for every Ethernet adapter manufactured. In order to send a packet to another computer, given the IP address of the other computer, it is first determined if the packet needs to be sent directly to the other computer or to the gateway. In either case, there is an IP address on the local network to which the packet must be sent. A table is maintained to allow the protocol driver to determine the MAC address corresponding to a particular IP address. If the table is empty, the MAC address is determined by sending an Ethernet broadcast packet to all devices on the local network asking the device with the desired IP address to answer with its MAC address. In this way, the table entry can be filled in. If no device answers, then the device is nonexistent or inoperative, and the packet cannot be sent.

IP addresses are arbitrary and can be allocated as desired provided that they don't conflict with other IP addresses. However, if they are to be used with the Internet, then they must be numbers that are assigned to your connection by proper authorities, generally by delegation via your service provider.

Each RCM2100 RabbitCore module has its own unique MAC address, which consists of the prefix 0090C2 followed by the code that appears on the label affixed to the RCM2100 module. For example, a MAC address might be 0090C2C002C0.

> **TIP:** You can always verify the MAC address on your board by running the sample program **DISPLAY_MAC.C** from the **SAMPLES\TCPIP** folder.

## 4.5 Dynamically Assigned Internet Addresses

In many instances, there are no fixed IP addresses. This is the case when, for example, you are assigned an IP address dynamically by your dial-up Internet service provider (ISP) or when you have a device that provides your IP addresses using the Dynamic Host Configuration Protocol (DHCP). The RCM2100 RabbitCore modules can use such IP addresses to send and receive packets on the Internet, but you must take into account that this IP address may only be valid for the duration of the call or for a period of time, and could be a private IP address that is not directly accessible to others on the Internet. These private addresses can be used to perform some Internet tasks such as sending e-mail or browsing the Web, but usually cannot be used to participate in conversations that originate elsewhere on the Internet. If you want to find out what this dynamically assigned IP address is, under Windows XP you can run the **ipconfig** program while you are connected and look at the interface used to connect to the Internet.

Many networks use private IP addresses that are assigned using DHCP. When your computer comes up, and periodically after that, it requests its networking information from a DHCP server. The DHCP server may try to give you the same address each time, but a fixed IP address is usually not guaranteed.

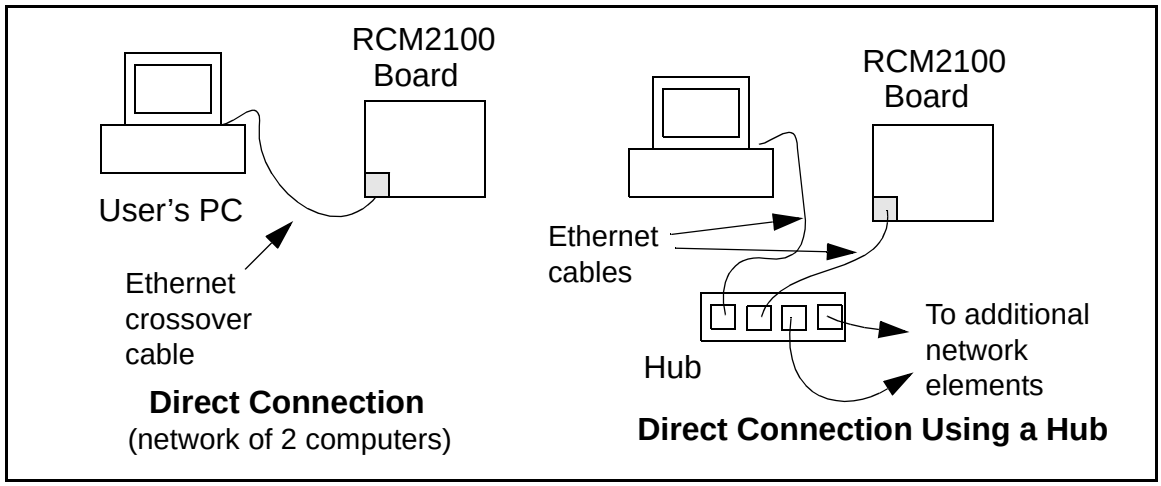
If you are not concerned about accessing the RCM2100 from the Internet, you can place the RCM2100 on the internal network using a private address assigned either statically or through DHCP.

## 4.6 Placing Your Device on the Network

In many corporate settings, users are isolated from the Internet by a firewall and/or a proxy server. These devices attempt to secure the company from unauthorized network traffic, and usually work by disallowing traffic that did not originate from inside the network. If you want users on the Internet to communicate with your RCM2100, you have several options. You can either place the RCM2100 directly on the Internet with a real Internet address or place it behind the firewall. If you place the RCM2100 behind the firewall, you need to configure the firewall to translate and forward packets from the Internet to the RCM2100.

## 4.7 Running TCP/IP Sample Programs

We have provided a number of sample programs demonstrating various uses of TCP/IP for networking embedded systems. These programs require you to connect your PC and the RCM2100 board together on the same network. This network can be a local private network (preferred for initial experimentation and debugging), or a connection via the Internet.



## 4.8 How to Set IP Addresses in the Sample Programs

With the introduction of Dynamic C 7.30 we have taken steps to make it easier to run many of our sample programs. Instead of the **MY_IP_ADDRESS** and other macros, you will see a **TCPCONFIG** macro. This macro tells Dynamic C to select your configuration from a list of default configurations. You will have three choices when you encounter a sample program with the **TCPCONFIG** macro.

1. You can replace the **TCPCONFIG** macro with individual **MY_IP_ADDRESS**, **MY_NETMASK**, **MY_GATEWAY**, and **MY_NAMESERVER** macros in each program.
2. You can leave **TCPCONFIG** at the usual default of 1, which will set the IP configurations to **10.10.6.100**, the netmask to **255.255.255.0**, and the nameserver and gateway to **10.10.6.1**. If you would like to change the default values, for example, to use an IP address of **10.1.1.2** for the RCM2100 board, and **10.1.1.1** for your PC, you can edit the values in the section that directly follows the "General Configuration" comment in the **TCP_CONFIG.LIB** library. You will find this library in the **LIB/TCPIP** directory.
3. You can create a **CUSTOM_CONFIG.LIB** library and use a **TCPCONFIG** value greater than 100. Instructions for doing this are at the beginning of the **TCP_CONFIG.LIB** file.

There are some other "standard" configurations for **TCPCONFIG** that let you select different features such as DHCP. Their values are documented at the top of the **TCP_CONFIG.LIB** library. More information is available in the ***Dynamic C TCP/IP User's Manual***.

### IP Addresses Before Dynamic C 7.30

Most of the sample programs such as shown in the example below use macros to define the IP address assigned to the board and the IP address of the gateway, if there is a gateway.

```
#define MY_IP_ADDRESS "10.10.6.170"
#define MY_NETMASK "255.255.255.0"
#define MY_GATEWAY "10.10.6.1"
#define MY_NAMESERVER "10.10.6.1"
```

In order to do a direct connection, the following IP addresses can be used for the RCM2100:

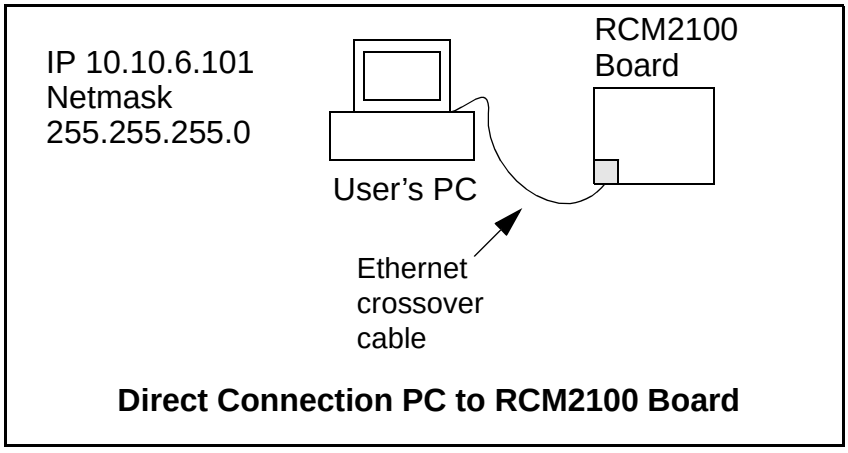
```
#define MY_IP_ADDRESS "10.1.1.2"
#define MY_NETMASK "255.255.255.0"
// #define MY_GATEWAY "10.10.6.1"
// #define MY_NAMESERVER "10.10.6.1"
```

In this case, the gateway and nameserver are not used, and are commented out. The IP address of the board is defined to be **10.1.1.2**. The IP address of your PC can be defined as **10.1.1.1**.

## 4.9 How to Set Up your Computer's IP Address for Direct Connect

When your computer is connected directly to the RCM2100 board via an Ethernet connection, you need to assign an IP address to your computer. To assign the PC the address `10.10.6.101` with the netmask `255.255.255.0`, do the following.

Click on **Start > Settings > Control Panel** to bring up the Control Panel, and then double-click the Network icon. Depending on which version of Windows you are using, look for the **TCP/IP Protocol**/**Network > Dial-Up Connections**/**Network** line or tab. Double-click on this line or select **Properties** or **Local Area Connection > Properties** to bring up the TCP/IP properties dialog box. You can edit the IP address and the subnet mask directly. (Disable "obtain an IP address automatically.") You may want to write down the existing values in case you have to restore them later. It is not necessary to edit the gateway address since the gateway is not used with direct connect.




**Direct Connection PC to RCM2100 Board**

## 4.10 Run the PINGME.C Sample Program

Connect the crossover cable from your computer's Ethernet port to the RCM2100 board's RJ-45 Ethernet connector. Open this sample program from the **SAMPLES\TCPIP\ICMP** folder, compile the program, and start it running under Dynamic C. When the program starts running, the green **LNK** light on the RCM2100 board should be on to indicate an Ethernet connection is made. (Note: If the **LNK** light does not light, you may not have a crossover cable, or if you are using a hub perhaps the power is off on the hub.)

The next step is to ping the board from your PC. This can be done by bringing up the MS-DOS window and running the pingme program:

```
ping 10.10.6.100
```

or by **Start > Run**

and typing the entry

```
ping 10.10.6.100
```

Notice that the red **ACT** light flashes on the RCM2100 board while the ping is taking place, and indicates the transfer of data. The ping routine will ping the board four times and write a summary message on the screen describing the operation.

## 4.11 Running More Sample Programs With Direct Connect

The sample programs discussed here are in the Dynamic C **SAMPLES\RCM2100** folder.

### 4.11.1 Sample Program: PINGLED.C

One of the RCM2100's most important features is the availability of the built-in Ethernet port. This program makes the simplest possible use of the network port by "pinging" a remote system and using LEDs to report the status of the ping attempt and its return.

#### Compile & Run Program

Open the **PINGLED.C** sample program. Press **F9** to compile and run the program.

Each time the program sends a ping to the remote address, LED DS2 on the Prototyping Board will flash. Each time a successful return from a ping attempt is received, LED DS3 will flash.

If the ping return is unsuccessful (i.e., the remote system does not exist or does not acknowledge the ping within the timeout period), DS3 will not flash.

With short ping times, as will be encountered in most micro-LAN and LAN settings, the two LEDs should flash almost in parallel as pings are sent and returned.

You can modify the **#define PING_DELAY** statement to change the amount of time between the outgoing pings.

## Program Description

For operation, network addresses must be correctly defined at the start of this program. The **TCPCONFIG 1** macro in the sample program provides default settings for **MY_IP_ADDRESS**, which is the address of the RCM2100 module, **MY_NETMASK**, and **MY_GATEWAY** (which needs to be defined if you wish to ping systems outside the local network). If you wish to ping systems using domain names instead of IP addresses, a valid DNS server address must be defined for **MY_NAMESERVER**. These TCP/IP settings can be changed as needed in the **TCP_CONFIG.LIB** library.

The IP address to be pinged is defined by **PING_WHO**. You will have to change this address and recompile the program to ping different addresses. (In most real-world applications, there should be some mechanism by which to dynamically define or select addresses.) This address may be defined as a numeric IP address. If a gateway to the Internet and a valid DNS server are specified, this definition may also be a fully-qualified domain name (such as "www.zworld.com").

The program first defines three functions to control the LEDs—one to initialize them, and then one each to drive the "ping out" and "ping in" LEDs.

The program begins by calling the LED initialization function **pingleds_setup( )**. More importantly, it then calls **sock_init( )**, which initializes the packet driver and the TCP manager using the compiler defaults. This function must always be called before any other TCP/IP functions.

The program then resolves the address to be pinged into a numeric value. using the library function **resolve()**. If the defined address is numeric, it converts the define string into truly numeric form. If the address is a domain name, the function queries the indicated DNS server to obtain the numeric address. (If the function is unable to resolve the address—if, for example, the numeric address is incomplete or badly formed, or the DNS server is unable to identify the domain name—the program will print a message to the screen and terminate.)

The program then begins an endless loop using **for(;;)**. Within this loop, the program executes the following steps:

1. Calls **tcp_tick()** to perform the basic housekeeping functions for the socket;
2. As a costatement, waits for the duration of **PING_DELAY** (defined by default as 500 ms or one-half second), issues a ping to the resolved address using the **_ping()** function, and flashes LED DS2;
3. As a second costatement, checks for a ping return using the **_chk_ping()** function. If the ping is successful, the costatement flashes LED DS3.

If you uncomment the **#VERBOSE** define near the beginning of the program, the ping return costatement will also print a message to the screen indicating each successful ping.

### 4.11.2 Sample Program: ETHCORE1.C

The RCM2100 modules with Ethernet ports can act as micro Web page servers, with dynamic interaction between the controller and the web pages. This sample program demonstrates how a Web page can be used to both monitor and control an RCM2100 module.

#### Compile & Run Program

Open the sample program **ETHCORE1.C**. Press **F9** to compile and run the program.

> **TIP:** This program will be more interesting to observe if LEDs DS4 and DS5 are installed on the Prototyping Board.

When the program starts, LEDs DS2, DS3 and DS5 will be lit, and DS4 will be dark. Open a Web browser and enter the IP address you defined for the RCM2100 module in the program in the address window. A page like that shown in Figure 8 should appear.

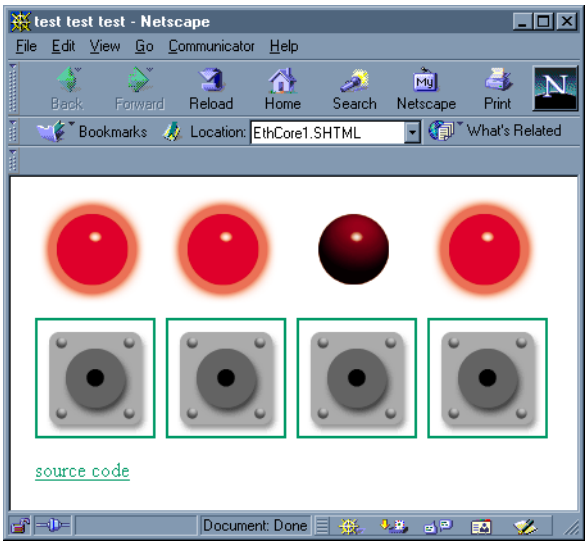

test test test - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print
Bookmarks Location: EthCore1.SHTML What's Related
source code
Document: Done

***Figure 8. Browser screen for Sample Program ETHCORE1.C***

Clicking on each of the button images in the browser window will toggle the state of the associated LED image, and will toggle the state of the corresponding LED on the Prototyping Board. Since the Web page is generated by the RCM2100 (using Dynamic HTML), the LED image and the corresponding LED's real state will always be in step.

#### Program Description

This program begins to show the range of applications for an Ethernet-enabled embedded system controller, so let's look closely at its operation.

As with **PINGLED.C**, several network addresses must be defined before this application can work. The **TCPCONFIG 1** macro in the sample program provides default settings for **MY_IP_ADDRESS**, which is the address of the RCM2100 module, **MY_NETMASK**, and **MY_GATEWAY** (which needs to be defined if you wish to reach the system from outside the local network). These TCP/IP settings can be changed as needed in the **TCP_CONFIG.LIB** library.

Generally, the other defined values may be left at their default settings. If you are operating the system behind a firewall or proxy and need to specify a host port for redirection, you should comment out the line reading:

```
#define REDIRECTHOST MY_IP_ADDRESS
```

Then uncomment the next line, which defines a specific redirection host and port:

```
#define REDIRECTHOST "my host.com:8080"
```

Be sure to enter the host port where indicated by **"my host.com:8080".**

This application creates dynamic HTML web pages on the fly. For simplicity, all of the Web page components—shell HTML, image GIFs, etc.—are imported into flash memory using the **#ximport** statements. It is also possible to read these files from other locations, including the onboard flash file system, but this application keeps things simple by loading all the components into working memory.

The program then defines four instances of an LED toggling function, which are basic CGI functions that swap the values "ledon.gif" and "ledoff.gif" as the contents of the **ledn** strings, and then force a reload of the web page to change the associated LED image. The physical LEDs on the Prototyping Board are turned on or off to match the **ledn** strings displayed on the Web page.

### 4.11.3 Additional Sample Programs

- **ETHCORE2.C**—This program takes anything that comes in on a port and sends it out Serial Port C. It uses SW2 as a signal that the connection should be closed, and PA0 as an indication that there is an open connection. You may change SW2 and PA0 to suit your application needs.

  Follow the instructions included with the sample program. Run the Telnet program on your PC (**Start > Run** `telnet 10.10.6.100`). As long as you have not modified the **TCPCONFIG 1** macro in the sample program, the IP address is 10.10.6.100 as shown; otherwise use the TCP/IP settings you entered in the **TCP_CONFIG.LIB** library. Each character you type will be printed in Dynamic C's **STDIO** window, indicating that the board is receiving the characters typed via TCP/IP.

- **LEDCONSOLE.C**—Demonstrates the features of **ZCONSOLE.LIB** command-oriented console library to control two LEDs on the Prototyping Board.

### 4.11.4 More Information

Refer to the ***Dynamic C TCP/IP User's Manual*** for complete details on the Dynamic C implementation of TCP/IP protocols.

## 4.12 Where Do I Go From Here?

**NOTE:** If you purchased your RCM2100 through a distributor or through a Z-World or Rabbit Semiconductor partner, contact the distributor or Z-World partner first for technical support.

If there are any problems at this point:

- Use the Dynamic C **Help** menu to get further assistance with Dynamic C.
- Check the Z-World/Rabbit Semiconductor Technical Bulletin Board at www.zworld.com/support/bb/.
- Use the Technical Support e-mail form at www.zworld.com/support/questionSubmit.shtml.

If the sample programs ran fine, you are now ready to go on.

Additional sample programs are described in the ***Dynamic C TCP/IP User's Manual***.

Please refer to the ***Dynamic C TCP/IP User's Manual*** to develop your own applications. ***An Introduction to TCP/IP*** provides background information on TCP/IP, and is available on the CD and on Z-World's Web site.

# NOTICE TO USERS

Z-WORLD PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE-SUPPORT DEVICES OR SYSTEMS UNLESS A SPECIFIC WRITTEN AGREEMENT REGARDING SUCH INTENDED USE IS ENTERED INTO BETWEEN THE CUSTOMER AND Z-WORLD PRIOR TO USE. Life-support devices or systems are devices or systems intended for surgical implantation into the body or to sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling and user's manual, can be reasonably expected to result in significant injury.

No complex software or hardware system is perfect. Bugs are always present in a system of any size. In order to prevent danger to life or property, it is the responsibility of the system designer to incorporate redundant protective mechanisms appropriate to the risk involved.

All Z-World products are 100 percent functionally tested. Additional testing may include visual quality control inspections or mechanical defects analyzer inspections. Specifications are based on characterization of tested sample units rather than testing over temperature and voltage of each unit. Z-World products may qualify components to operate within a range of parameters that is different from the manufacturer's recommended range. This strategy is believed to be more economical and effective. Additional testing or burn-in of an individual unit is available by special arrangement.

# INDEX


## A

additional information
  online documentation ..... 5
  references ..... 5

## D

description ..... 1
Development Kit ..... 7
Dynamic C ..... 4, 17
  add-on modules ..... 29
  features ..... 17
  standard features ..... 18
    debugging ..... 18
  telephone-based technical support ..... 29
  upgrades and patches ..... 29
  USB port settings ..... 15

## E

Ethernet cables ..... 31
Ethernet connections ..... 31, 33
  10Base-T ..... 33
  10Base-T Ethernet card .... 31
  additional resources ..... 46
  direct connection ..... 33
  Ethernet cables ..... 33
  Ethernet hub ..... 31
  IP addresses ..... 33, 35
  steps ..... 31, 32

## F

features
  Prototyping Board ..... 8, 9, 10
  RCM2100 ..... 1

## H

hardware connections ..... 11
  install RCM2100 on Prototyping Board ..... 12
  power supply ..... 14
  programming cable ..... 13
hardware reset ..... 14

## I

IP addresses ..... 35
  how to set in sample programs ..... 40
  how to set PC IP address ... 41

## M

MAC addresses ..... 36
models
  factory versions ..... 1, 2

## P

pinout
  RCM2100 ..... 3
power supply
  connections ..... 14
programming cable
  RCM2100 connections ..... 13
Prototyping Board ..... 8
  features ..... 8, 9, 10
  mounting RCM2100 ..... 12

## R

RCM2100
  mounting on Prototyping Board ..... 12
reset ..... 14

## S

sample programs
  getting to know the RCM2100
    EXTSRAM.C ..... 21
    FLASHLED.C ..... 21, 26
    FLASHLED2.C ..... 21
    FLASHLEDS.C ..... 22, 27
    FLASHLEDS2.C ..... 22
    KEYLCD2.C ..... 22
    LCD_DEMO.C ..... 23
    SWTEST.C ..... 23
    TOGGLELED.C ..... 23, 28
  how to run TCP/IP sample programs ..... 39, 40
  how to set IP address ..... 40
  PONG.C ..... 15
  serial communication
    CORE_FLOWCONTROL.C ..... 24
    CORE_PARITY.C ..... 24
    MASTER2.C ..... 25
    SLAVE2.C ..... 25
  TCP/IP
    DISPLAY_MAC.C ..... 36
    ETHCORE1.C ..... 44
    ETHCORE2.C ..... 45
    LEDCONSOLE.C ..... 45
    PINGLED.C ..... 42
    PINGME.C ..... 42
specifications
  physical and electrical ..... 2

## T

TCP/IP primer ..... 33
technical support ..... 16

## U

USB/serial port converter ..... 13
  Dynamic C settings ..... 15

# SCHEMATICS

## 090-0114 RCM2100 Schematic

www.rabbitsemiconductor.com/documentation/schemat/090-0114.pdf

## 090-0116 RCM2100 Prototyping Board Schematic

www.rabbitsemiconductor.com/documentation/schemat/090-0116.pdf

## 090-0128 Programming Cable Schematic

www.rabbitsemiconductor.com/documentation/schemat/090-0128.pdf

The schematics included with the printed manual were the latest revisions available at the time the manual was last revised. The online versions of the manual contain links to the latest revised schematic on the Web site. You may also use the URL information provided above to access the latest schematics directly.